POLITECNICO DI TORINO
Repository ISTITUZIONALE

Scheduling Traffic for Maximum Switch Lifetime in Optical Data Center Fabrics

Original
Scheduling Traffic for Maximum Switch Lifetime in Optical Data Center Fabrics / Bianco, Andrea; Giaccone, Paolo; Ricca, Marco. - In: COMPUTER NETWORKS. - ISSN 1389-1286. - STAMPA. - 105:(2016), pp. 75-88. [10.1016/j.comnet.2016.05.002]

Availability:
This version is available at: 11583/2641784 since: 2016-07-12T11:59:38Z

Publisher:
Elsevier

Published
DOI:10.1016/j.comnet.2016.05.002

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

18 April 2024

Scheduling Traffic for Maximum Switch Lifetime in Optical Data Center Fabrics

Andrea Bianco, Paolo Giaccone, and Marco Ricca

Department of Electronics and Telecommunications, Politecnico di Torino, Italy
E-mail: andrea.bianco@polito.it, paolo.giaccone@polito.it, marco.ricca@polito.it

Abstract

Optical switching is a promising technology to scale the performance of data centers. We address two scenarios: hybrid data centers, in which an optical circuit switching network operates in parallel to an electronic packet switching network that interconnect the servers, and full optical data centers, in which all the traffic is switched in the optical domain. The traffic on the optical network is scheduled proactively by configuring the optical fabric according to a precomputed sequence of configurations. We consider that fast optical circuit switching is implemented through micro-mechanical devices (like MEMS) at each crosspoint, whose lifetime is limited by a maximum number of configuration cycles, due to the experienced mechanical fatigue. We consider the traffic scheduling problem on the optical network and address specifically the problem of minimizing the number of reconfigurations at each crosspoint, in order to maximize the MEMS lifetime, while maximizing the throughput. We propose a family of scheduling algorithms and discuss the achievable tradeoff between lifetime, throughput and computational complexity, throughout a set of extensive simulations and theoretical results.

Keywords: Optical data centers, MEMS-based optical fabrics, frame-based packet scheduling.

1. Introduction

In the last few years the design of high-performance data centers has been evolving toward hybrid design approaches, in which an Optical Circuit Switching (OCS) network complements the operation of a classical Electronic Packet Switching (EPS) network [1, 2, 3, 4, 5, 6, 7]. This trend has been mainly driven by two factors.

First, a large amount of traffic within a datacenter is generated by few long-lived flows, whose demand is quite stable if measured at a proper timescale [8, 9]; thus, the network-wide traffic demand can be estimated [1, 10]. Due to its overhead, circuit switching is adopted only for such long-lived flows, that can be routed across specific and dedicated network paths. Notably, circuit switching improves the communication efficiency, thanks to the complete control of the achievable performance and of the induced network congestion. Differently from the Internet scenario, circuit switching can be efficiently supported in data centers. Indeed, a centralized traffic arbitration is feasible thanks to the fact that servers and the network devices are physically collocated ("under the same roof") and under the same administrative domain.

Second, optical switching provides the most efficient way to forward traffic in terms of power and bandwidth, but suffers from inherent reconfiguration latency (in the order of ms or μs) that make it convenient only for circuit switching, and not for packet switching. Thus, switching occurs at *flow* level with temporal dynamics much slower than packet level (occurring with temporal dynamics of few nanoseconds). Two main approaches can be adopted to integrate OCS in data centers. In a less disruptive solution, an OCS network is adopted to connect Top-of-Rack (ToR) switches to create dedicated bypass connections that adapt to the current traffic conditions. Thus each ToR switch is "dual-homed" to a traditional EPS network and to an OCS network. The main idea is to distribute the traffic among the two networks, based on its level of predictability. E.g., elephant flows are sent through the OCS whereas mice flows are sent to the EPS. Some commercial products are already exploiting this hybrid approach [11]. Recently, [12] has provided an economic analysis of their proposed hybrid data center and shown the lower costs with respect to full electronic data centers, given the same capacity. A more disruptive solution is to adopt only OCS to connect the ToR switches, as discussed in [13, 14]. This may pose extra burn on the optical switching devices due to the possibly higher frequency by which the OCS must be configured.

Regarding the control of OCS networks, proactive traffic scheduling has been proposed [7, 10, 15, 16] to switch traffic based on a-priori knowledge of the traffic demands between ToR switches. Periodically, the scheduler computes the new traffic demands and compute the future switching configurations for the OCS network, defined by (i) the temporal sequence of flows that each ToR switch must send to the OCS network, (ii) and the corresponding routing path in the OCS network topology. In other terms, the scheduler must compute the temporal sequence (called "frame") of switching configurations and the routing paths compatible with the available bandwidth in the OCS network.

To build the OCS network, one of the most common ap-

proach is to exploit optical MEMS, as proposed in [5, 7, 10, 17, 18]. The MEMS technology [19, 20], nowadays mature, is based on the mechanical movement of micro mirrors which deflect the laser beam. The switching performance of the MEMS is limited, mainly due to two factors. First, the maximum switching speed is limited to ms-μs [18], which is compatible with the temporal dynamics of the relatively slow OCS. For example, [7] designs an hybrid data center exploiting a MEMS-based switch Mordia, previously proposed by [10]. This optical switch manages 24 ports running at 100 Gbps, with a reconfiguration delay of around 10 μs.

Second, the lifetime of the MEMS is limited, indeed vendors suggest a maximum number of switching cycles, which can reach at most 10^9 switching cycles for the most recent technologies [17, 21]. This is due to electrical or mechanical overstress and applies at each individual optical mirror [22, 23]. The peculiar stress reason (e.g., creep, contact wear, friction) is significantly articulated and depends on the actual technology, materials and design adopted in the MEMS [24]. Thus, we will assume that the maximum number of switching cycles refers *to each single micro mirror*.

The aim of our work is to schedule the traffic in order to minimize the number of reconfigurations (thus, the mechanical fatigue) experienced by each single micro mirror in the OCS network, thus maximizing the lifetime of the optical devices, while guaranteeing high throughput. Intuitively, our approach is based on changing the switching configuration in a "lazy" way, i.e., trying to keep the switching configuration as similar as possible in consecutive times in order to minimize the number of micro mirrors that are moved. We expect that by decreasing the number of reconfigurations by a factor α, we can either improve the lifetime of the OCS devices by the same factor, or, given the same lifetime, we can increase by α the number of flows that can be optically switched. We can also expect an higher reliability for larger values of α.

Our work applies to the OCS architecture for hybrid data centers proposed by [7], to which we refer for all the implementation issues. We propose a scheduling approach which can run in the implementation of [7] and can obtain the experimental performance shown there. Notably our approach is able to maximize the lifetime of the MEMS-based OCS network, which is not taken into account in previous works, as discussed in Sec. 6. Our approach is general and applies to any MEMS-based full optical switch. Notably, it can be integrated into existing traffic schedulers at a negligible additional cost in term of computational complexity, greatly compensated by the improvement in the MEMS lifetime.

The main contributions of our work are the following:

- First, we propose a fatigue-aware scheduling framework, based on different combinations of algorithms, aimed at maximizing throughput and minimizing the fatigue, at the same time, in order to maximize the lifetime of the OCS network while preserving the performance.
- Second, we consider a set of random traffic patterns and we assess the performance analytically under different scheduling algorithms.

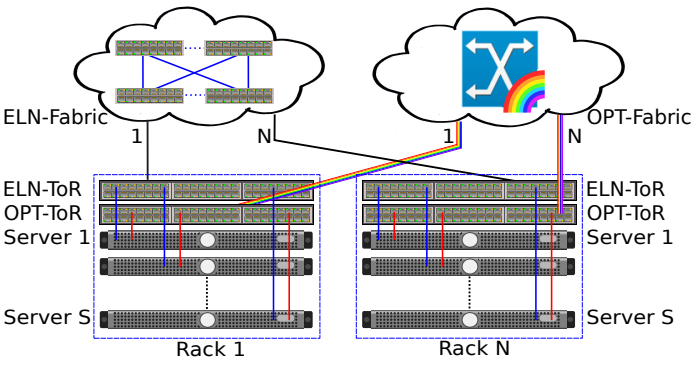

Figure 1: Hybrid data center architecture based on N ToR racks, each of them hosting S servers. Each server is connected to both an ELN-ToR switch and an OPT-ToR switch. The former switch accesses an electronic packet switch network (ELN-fabric) and the latter accesses an optical circuit switch network (OPT-fabric). Each fabric has N bidirectional ports, one for each rack.

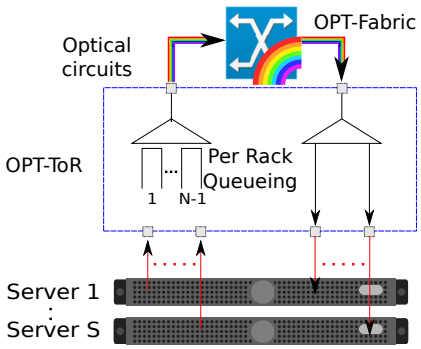

Figure 2: Datapath in the OCS-ToR switch for ingress and egress flows. Incoming flows are buffered in electronic memories organized with one queue for each destination rack.

- Third, we validate the approach on a wide set of traffic patterns through simulation and we highlight the algorithm achieving the best tradeoff between throughput, fabric lifetime and computational complexity.

The paper is organized as follows. Sec. 2 defines the scheduling problem and in Sec. 3 we which is propose our fatigue-aware scheduling approach, whose performance is later investigated in terms of fatigue and throughput both analytically (Sec. 4) and by simulation (Sec. 5). Sec. 6 is devoted to discuss the related work. Finally, Sec. 7 draws the conclusions of our work.

2. Problem Definition

We consider a data center with physical servers distributed across N racks, as shown in Fig. 1, where each rack hosts S servers. Typical values are $S = 40$ for a standard 42U 19-inches rack, but can reach also some hundreds in the case of blade servers. In total, the data center comprises NS servers. In the considered hybrid scenario, each rack is equipped with two Top-of-Rack (ToR) switches. The first ToR is denoted as ELN-ToR (Electronic ToR) and is connected to the EPS network. Despite the complex switching network (e.g., multi-layer, hierarchical) required to interconnect the ELN-ToR switches, we

abstract it as a single EPS fabric, denoted for simplicity as *ELN-fabric*. Similarly, the second ToR switch in the rack is denoted as OPT-ToR (Optical ToR) and is connected to an OCS network, denoted as *OPT-fabric*. Note that both ELN-ToR and OPT-ToR are electronic switches, but in particular the OPT-ToR is equipped with the optical interfaces needed to access the OPT-fabric. The number of bidirectional ports in each fabric is N, equal to the number of racks. Each server is connected to both ToR switches with two dedicated interfaces and it is responsible of classifying the traffic and routing it to the proper ToR switch. We refer to the similar architecture presented in [7, 12] for the implementation details of the traffic management within each server.

In our work we focus only on the traffic sent to OPT-fabric, which is bufferless by construction and thus can be modeled as an $N \times N$ crossbar interconnecting the OPT-ToR switches. To avoid the performance degradation due to head-of-line blocking, each OPT-ToR is equipped with input queues organized with a Virtual Output Queue (VOQ) structure, i.e. one FIFO queue is present for each OPT-ToR switch, or equivalently for each destination rack, as shown in Fig. 2. In total, $(N-1)$ per-rack queues are available in each OPT-ToR switch. Let VOQ_{ij} be the queue storing the data of the flows from OPT-ToR switch (or rack) i and destined to OPT-ToR switch (or rack) j.

A traffic scheduler, denoted as *OPT-scheduler*, determines the switching configuration of the OPT-fabric, by specifying the connections among the OPT-ToR switches and their duration. Due to the lack of the internal buffers in the OPT-fabric, at most one traffic flow must be transferred from each rack and can be destined to the same rack, as considered in all works on hybrid data centers. Thus, the OPT-fabric configuration can be modeled as a *matching* in a bipartite graph with $2N$ nodes, where the N left-most nodes correspond to the input ports of the OPT-fabric and N right-most nodes corresponds to its output ports. An edge in the matching from port i to port j corresponds to the traffic flows transferred from ToR-switch/rack i to ToR-switch/rack j. A matching can be represented by an $N \times N$ binary matrix $M = [m_{ij}]$, denoted as *matching matrix*, in which $m_{ij} = 1$ iff ToR-switch/rack i is connected to ToR-switch/rack j, and at most one element is set to 1 in each row and in each column: $\sum_{k=1}^{N} m_{ik} \leq 1$, $\sum_{k=1}^{N} m_{kj} \leq 1$, $\forall i, j$. The set of all matching matrices is denoted by $\mathcal{M}$. A matching M is *complete* if exactly one element is set to 1 in each row and in each column, and it is represented by a permutation matrix; otherwise, it is defined as *incomplete*. Finally, a matching is *non-null* if at least one element is set to 1: $\sum_{i,j} m_{ij} \geq 1$.

We assume that OPT-fabric runs in a synchronous way, where the reconfigurations occur at multiples of a basic timeslot of fixed duration. The data in each flow is divided into fixed-size *chunks*, whose size corresponds to the data transferred in each timeslot. Thus, by construction, during each timeslot, one chunk of a flow is transferred across the OPT-fabric based on a matching computed by the OPT-scheduler. Whenever a new matching is adopted, a reconfiguration latency is experienced, during which no chunk can be transferred across the fabric. This latency depends on the technology adopted in the OPT-fabric, and can be order of ms-μs in the case of MEMs. The timeslot duration is chosen large enough to guarantee a minimum throughput. For example, to guarantee at least 95% throughput for a reconfiguration latency of 1 ms, the timeslot duration must be chosen larger than 19 ms. This corresponds to chunks of at least 95 MB for 100 Gbit/s optical interfaces. In the following, without loss of generality, we will deliberately neglect the reconfiguration latency in our model, since it can be managed a-priori by dimensioning the timeslot duration properly. We refer to [7] for the details of the implementation of the synchronous reconfiguration system. Notably, [7] proposes a host-based control protocol, which injects the data to the OPT-fabric based on 802.1Qbb Priority Flow Control. The correct timing is crucial since one must take into account all the possible control latencies due to many practical issues, e.g.: the reconfiguration delays for the MEMS, the processing delay of the pause commands, the packets under transmission and the lock time of the lasers. Furthermore, in the case of TCP flows, the scheduling decision affects the behavior of the window-based protocol. The most notable effects, highlighted by [7], are due to the periodic starting and pausing process on the TCP flows and to the TCP segmentation offloading in the server network interface. Note that ACKs are sent back through the ELN-fabric to avoid pauses. In conclusion, all the proposed implementation solutions adopted in [7] can be applied to the data center architecture considered in our work.

2.1. Frame-based Traffic Scheduling

The aim of OPT-scheduler is to minimize the overall fatigue experienced by OPT-fabric on a time interval, while maximizing throughput. We assume that the scheduler operates on a frame basis, which is a well-known approach in EPS switches since [25]. The scheduler samples the state of the VOQs across all the racks at the beginning of a fixed *sampling period*, that lasts T timeslots; i.e. the queues are sampled at timeslot $t = nT$, for any $n \in \mathbb{N}$. Then it computes a *frame* $\mathcal{F}$, i.e. a sequence of matchings, in order to empty the input queues before the next sampling period. Finally, the OPT-fabric is configured according to $\mathcal{F}$ to serve these chunks during the following T timeslots, i.e. for any $t \in [nT, (n+1)T)$. If some queues are not empty at the end of the frame (i.e., at $t = (n+1)T$), the residual chunks are kept in the queues and will be served in one of the subsequent sampling periods. Indeed, the queue occupancy sampled at $t = nT$ is given by the sum of newly arrived chunks during the last frame and the residual chunks. According to standard approaches, the two phases of computing the frame and serving the chunks can be pipelined in subsequent scheduling periods; this allows to amortize the time to compute a new frame on the whole sampling period.

We introduce a matrix operator which returns the maximum row and column sum of a generic matrix $X = [x_{ij}]$ of size $N \times N$:

$$h(X) = \max \left\{ \max_{j=1\ldots N} \sum_{i=1}^{N} x_{ij}, \ \max_{i=1\ldots N} \sum_{j=1}^{N} x_{ij} \right\}$$

Let $A_n = [a_{ij}(n)]$ be the *arrival matrix*, where $a_{ij}(n)$ is the number of chunks arrived at VOQ_{ij} during the whole nth sam-

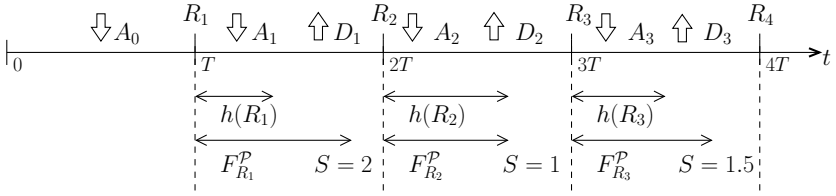

Figure 3: Temporal evolution of the frame-based traffic scheduling with arrivals and departures during 4 sampling periods.

pling period. If the arrival process at the OPT-ToR switch is stationary, we can define $\hat{\Lambda} = [\hat{\lambda}_{ij}]$ as the average *traffic matrix* measured in Gbps, where λ_{ij} is the offered load to VOQ_{ij}. Given a transmission rate of B Gbps at the OPT-fabric ports, we can compute the corresponding *normalized load matrix* $\Lambda = [\lambda_{ij}]$ where $\lambda_{ij} = \hat{\lambda}_{ij}/B \in [0, 1]$ is the normalized load at VOQ_{ij} and it holds: $\lambda_{ij} = \mathbb{E}[a_{ij}]/T$. Let Λ be defined *admissible* whenever no input/output port of the OPT-fabric is oversubscribed, i.e. whenever $h(\Lambda) < 1$. We are not taking into account the reconfiguration latency, since the throughput reduction due to it is a-priori known and can be compensated with a large enough timeslot duration.

As toy example to illustrate the notation, consider a 2×2 OPT-fabric with links at $B = 100$ Gbps fed by traffic matrix $\hat{\Lambda} = \left(\begin{smallmatrix} 40 & 50 \\ 10 & 30 \end{smallmatrix}\right)$ Gbps, thus the normalized load matrix is $\Lambda = \left(\begin{smallmatrix} 0.4 & 0.5 \\ 0.1 & 0.3 \end{smallmatrix}\right)$. Here $h(\Lambda) = 0.9$ and thus Λ is admissible.

Let $R_n = [r_{ij}(n)]$ be the *request matrix*, where $r_{ij}(n)$ is the number of chunks enqueued at VOQ_{ij}, sampled at the beginning to the nth sampling period, i.e. at timeslot $t = nT$. Finally, let $D_n = [d_{ij}(n)]$ be the *departure matrix*, computed by the scheduler based on R_n (and obliviously from A_n since unknown at the beginning of the sampling period), where $d_{ij}(n)$ is the number of chunks served from VOQ_{ij} during the nth sampling period. According to the above definitions: $R_{n+1} = R_n - D_n + A_n$. Fig. 3 shows an example of system evolution during 4 consecutive sampling periods (i.e., $n \in \{0, 3\}$ and $t \in [0, 4T)$).

We can complete the above toy example by fixing a sampling period of $T = 10$ timeslots. If we assume $R_n = \left(\begin{smallmatrix} 4 & 8 \\ 5 & 1 \end{smallmatrix}\right)$ chunks, then the scheduler computes a departure matrix $D_n = \left(\begin{smallmatrix} 4 & 6 \\ 5 & 1 \end{smallmatrix}\right)$ chunks to be applied during the nth sampling period. In the meanwhile, an arrival matrix possible with Λ could be $A_n = \left(\begin{smallmatrix} 5 & 0 \\ 2 & 4 \end{smallmatrix}\right)$ chunks. Thus, at the beginning of the next sampling period, the new request matrix will be $R_{n+1} = \left(\begin{smallmatrix} 5 & 2 \\ 2 & 4 \end{smallmatrix}\right)$ chunks.

Thanks to the Birkhoff-von Neumann theorem [26], the minimum frame duration T_R in timeslots to serve all the chunks in[1] R is $T_R = h(R)$. Thus, when $T_R > T$, it is impossible to serve all chunks in R during the sampling period, whereas when $T_R \leq T$ there exists at least one sequence of matchings able to empty all the chunks in R and we say that R is T-compatible. Equivalently, if we define the *frame maximum load* as $\rho_R = T_R/T$, the request matrix R is *T-compatible* when $\rho_R \leq 1$. Due to the stochastic nature of the arrivals to the OPT-ToR switch, the admissibility of Λ does not imply the T-compatibility of R. Indeed, even if $h(\Lambda) = \epsilon$ for some small $\epsilon > 0$, it can occur that $\rho_R > 1$ with small probability due to a continuous sequence of chunks directed to the same output port. But [27] showed that for large enough T, precisely when $T = \theta(\log N)$, being N the size of the OPT-ToR switch, the admissibility of Λ implies the T-compatibility of R with high probability.

The main advantage of all the above definitions is that the concept of T-compatible request matrix is well defined also for non-stationary and/or non-admissible and/or correlated arrival processes.

In the above toy example, $T_{R_n} = 12$ and thus R_n is not 10-compatible since it is not possible to transfer all the enqueued chunks in $T = 10$ timeslots ($\rho_{R_n} = 1.2$ in this case). Instead, $T_{R_{n+1}} = 9$ and thus R_{n+1} is instead 10-compatible ($\rho_{R_{n+1}} = 0.9$).

The departure matrix D_n must be defined according to a sequence of matchings as follows. The frame $\mathcal{F}^{\mathcal{P}}_R$ computed by a specific scheduler $\mathcal{P}$ on the request matrix R is defined as an ordered sequence of K distinct and non-null matchings: $\mathcal{F}^{\mathcal{P}}_R = \{(M^k, \phi_k)\}_{k=1}^K$, with $M^k \in \mathcal{M}$ and $\phi_k \in \mathbb{N}$ is the number of consecutive timeslots in which matching M^k is used to configure the OPT-fabric. To serve all chunks in R, it must hold[2]:

$$R = \sum_{k=1}^{K} \phi_k M^k \tag{1}$$

Let $F^{\mathcal{P}}_R = \sum_{k=1}^K \phi_k$ be the frame duration, i.e. the total number of timeslots to transfer the chunks and to empty the queues. Note that T is fixed, whereas $F^{\mathcal{P}}_R$ varies with R. A T-compatible request matrix R is said to be *sustainable* by scheduling algorithm $\mathcal{P}$ if during a sampling period *all* the chunks in the request matrix are transferred, i.e. if

$$F^{\mathcal{P}}_R \leq T \tag{2}$$

Note that a T-compatible request matrix could be non sustainable if the scheduling algorithm was not able to serve all the chunks within T timeslots (even if in theory it could be possible by an optimal algorithm).

In general, it can also happen that $F^{\mathcal{P}}_R \geq T_R$ and we define the *frame-expansion factor* S (with $S \geq 1$) as:

$$S = F^{\mathcal{P}}_R / T_R \tag{3}$$

where S depends from both R and $\mathcal{P}$, but we omitted them to preserve concise notation. Combining (2) and (3), R is sustainable if $S \leq 1$. If R_n happens to be sustainable for any n

[1] For the sake of notation, we omit n from the notation when not necessary

[2] Thanks to the definition of the matching matrix, which may be incomplete, (1) holds with the equal sign.

under some scheduling policy $\mathcal{P}$, thus $1/S$ is defined as the normalized *maximum sustainable load* under $\mathcal{P}$. Two possible approaches can be devised to compensate a maximum sustainable load less than one. As a first solution, a rate-limiting scheme can be directly implemented at the servers, ensuring that no more than $1/S$ fraction of traffic is sent to the OPT-fabric from any ToR switch. As a second solution, it is possible to compensate with a bandwidth over-provisioning on the optical fabric by a factor $1/S$. The additional cost of such solution is expected to be limited in case of full optical fabric, also thanks to the fact that the switching capabilities of MEMS are independent of the data rate of the optical signal.

Fig. 3 shows an example of S values based on the minimum frame duration T_{R_n} and the actual frame duration $F^{\mathcal{P}}_{R_n}$. In the first period, only arrivals and no departures occur (since $R_0 = 0$). In the second period, the frame duration is twice the minimum one ($S = 2$), but R_1 is still sustainable. The same occurs for the two subsequent frames. In this example, since all the request matrices are sustainable, all the enqueued chunks at the beginning of each sampling period are served and thus each new request matrix depends only on just the arrivals during the previous sampling period.

We highlight the self-adaptive property of the proposed frame scheduler approach. It does not require any a-priori knowledge of the traffic matrix, since at each new sampling period the updated state of the queue is considered to compute the new frame. This is different from [7], which assumes to know in advance the traffic matrix, relying on standard methods (as the ones proposed by [1]) to estimate it.

2.2. Fatigue model for the OPT-fabric

In order to maximize the lifetime of the OPT-fabric, we minimize the variations among matchings in consecutive timeslots. More precisely, the *fatigue cost* during some period of time is defined as the number of variations at edge level occurring in the sequence of matchings, i.e. the number of movements occurring at single micro-mirror level. If input i is connected to output j at timeslot t and then becomes connected to a different output $k \neq j$ at timeslot $t+1$, two variations arise: the first one to remove the connection from input i to output j and the second one to setup the new connection from input i to output k. The fatigue cost in this case is assumed to be 2 since this event corresponds to change the position of two micro mirrors. In the case a single micro-mirror was associated to each input port, the change of destination for one port would cost just one variation, instead of two as considered in our work. Since in this case the fatigue cost would be half than the one considered in our work, the corresponding optimization problem would be the same as in our paper.

The fatigue cost of a new matching is obtained as the sum of variations required to remove all connections selected in the previous timeslot and not selected in the current timeslot, plus the cost required to setup all connections selected in the current timeslot and not selected in the previous one. This implies that the fatigue cost between consecutive matchings is always between zero (no change) and $2N$ (complete change). Let $E(M^h, M^k)$ be defined as the total fatigue cost occurred to modify matching $M^h = [m^h_{ij}]$ into $M^k = [m^k_{ij}]$. It can be formally computed as follows:

$$E(M^h, M^k) = \sum_{i=1}^{N} \sum_{j=1}^{N} |m^h_{ij} - m^k_{ij}|$$

By construction, $E(M^h, M^k) = E(M^k, M^h)$.

The total fatigue cost needed to reconfigure the OPT-fabric, according to frame $\mathcal{F}^{\mathcal{P}}_R$, is:

$$E(\mathcal{F}^{\mathcal{P}}_R) = \sum_{k=1}^{K-1} E(M^k, M^{k+1})$$

Note that $E(\mathcal{F}^{\mathcal{P}}_R)$ is independent of the values of ϕ_k, since the configuration contribution due to consecutive identical matchings is null.

Table 1: Throughput and fatigue for constant uniform request matrix

Frame	$F^{\mathcal{P}}_R$	S	Maximum sustainable load	Fatigue cost per frame
$\mathcal{F}_1$	T_R	1	1	$2uN^2$
$\mathcal{F}_2$	T_R	1	1	$2N^2$
$\mathcal{F}_3$	NT_R	N	$1/N$	$2N^2$

2.3. A Toy Example

To understand the possible tradeoff between throughput and fatigue, we consider the case of a constant uniform request matrix R, where $r_{ij} = u$, $\forall i, j$, and u is a fixed positive integer ≥ 1; in this case, $T_R = Nu$. Let $D^k = [d^k_{ij}] \in \mathcal{M}$ be the permutation matrix corresponding to the i-th diagonal, for $i = 1 \dots N$. As example, in a 3×3 OPT-fabric,

$$D^1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},\ D^2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix},\ D^3 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Let us consider three possible frames, all of them satisfying (1):

- $\mathcal{F}_1 = \{(D^1, 1), \dots, (D^N, 1), \dots, (D^1, 1), \dots, (D^N, 1)\}$: the matchings are cyclically selected among all the N diagonals in a round robin fashion, keeping each matching for one timeslot.
- $\mathcal{F}_2 = \{(D^1, u), \dots, (D^N, u)\}$: the matchings are cyclically selected among all the N diagonals in a round robin fashion, keeping each matching for u consecutive timeslot.
- $\mathcal{F}_3 = \{(U^{11}, u), \dots, (U^{1N}, u), \dots, (U^{N1}, u), \dots, (U^{NN}, u)\}$: where U^{ij} is a matching with only one edge, from input i to output j.

Table 1 reports the corresponding sustainable load and configuration cost per frame. From the throughput point of view, only $\mathcal{F}_1$ and $\mathcal{F}_2$ are optimal. From the fatigue point of view, instead,

$\mathcal{F}_2$ and $\mathcal{F}_3$ are optimal, thanks to the fact that they force the use of the same matching in consecutive timeslots. As a conclusion, under the considered traffic matrix, $\mathcal{F}_2$ is the best scheduling decision, achieving optimality in terms of both fatigue and throughput.

3. Fatigue-Aware Frame Scheduling

We aim at finding scheduling policies that maximize throughput (i.e., achieving the maximum sustainable load) and minimize the fatigue cost (i.e., the variations between consecutive identical matchings are minimum), under a generic request matrix.

Our fatigue-aware frame-scheduling problem can be modeled as a two-objective optimization problem: define a frame that minimizes the fatigue costs whilst maximizing the throughput (or, equivalently, minimizing the frame duration $F_R^{\mathcal{P}}$). We propose to solve this problem in two phases, each of them aimed at a specific objective:

- **Matching selection**: given the request matrix R, define an algorithm $\mathcal{P}$ that computes an *unordered* frame $\mathcal{U}_R^{\mathcal{P}} = \{M^k, \phi_k\}_{k=1}^K$ such that condition (1) is satisfied and the corresponding frame duration is minimized. The objective is to serve all the enqueued chunks in R to maximize throughput.
- **Frame sorting**: compute the final frame $\mathcal{F}_R^{\mathcal{P}}$ by ordering $\mathcal{U}_R^{\mathcal{P}}$ to minimize the switching fatigue cost.

In the following sections, we will discuss each phase separately.

3.1. Matching Selection

We consider five different algorithms for the matching selection. The first four are iterative algorithms, exploiting the same decomposition algorithm *Gen-DEC* as template, whose pseudo-code is reported in Fig. 4. At each iteration of *Gen-DEC*, a specific algorithm $\Omega(R)$ computes a matching matrix M on R (line 3). Each of the four algorithm is characterized by a specific $\Omega(R)$. Then, the value of the minimum element in R among those selected by the matching matrix M (line 4) is subtracted from all selected elements in R (line 5), and a residual request matrix is obtained (line 6). The process iterates until R becomes empty. Since, at each iteration, at least one element (at most N elements) of R becomes zero, N^2 iterations are needed in the worst case to fully schedule R.

We considered in this paper five frame decomposition algorithms:

- **BvN**: a *Gen-DEC* based algorithm, exploiting the Birkhoff-von Neumann decomposition [26] on R, satisfying condition (1). Thus, at each iteration $\Omega(R)$ is a MSM (Maximum Size Matching) on R, i.e. the matching with the largest number of edges corresponding to non-null elements of R. The MSM algorithm complexity is $O(N^{2.5})$. Differently from the original algorithm in [26], we are not running the initial augmentation phase on R to get a new matrix with all the rows and columns summing to the same value. Thus, our implementation is not provably optimal in terms of minimum frame duration. Nevertheless, for most of the scenarios considered later, the algorithm will be able to find the minimum frame (equal to T_R) and thus achieving the maximum sustainable load. The overall computational complexity is $O(N^{4.5})$.

Gen-DEC (*Input*: R; *Output*: $\mathcal{U}_R$)
1. $\mathcal{U}_R = \emptyset, k = 1, R(k) = R$ // initialize
2. **while** $R(k) \neq 0$ // while $R(k)$ is not completely zero
3. $M^k = \Omega(R(k))$ // find a matching
   // find the minimum value of R corresponding to M^k
4. $\phi_k = \min_{1 \leq i,j \leq N}\{m_{ij}^k r_{ij}(k) | r_{ij}(k) > 0\}$
5. $R(k+1) = R(k) - \phi_k M^k$ // subtract
6. $\mathcal{U}_R = \mathcal{U}_R \cup \{(M^k, \phi_k)\}$ // frame update
7. $k = k + 1$ // start a new iteration

Figure 4: Pseudocode of Gen-DEC scheduling algorithm

- **GMax**: a *Gen-DEC* based algorithm, where $\Omega(R)$ is a greedy maximum weight matching on R. I.e., at each iteration, the algorithm to compute $\Omega(R)$ selects the largest element in R, then it removes the corresponding row and column from R, and repeats the process until all the rows and columns in R have been considered. The complexity of each iteration is $O(N^2 \log N)$, due mainly to the initial sorting of the N^2 values in R; hence, the overall computational complexity is $O(N^4 \log N)$.
- **GExa**: a *Gen-DEC* based algorithm. $\Omega(R)$ is a maximal size matching with the constraint that a queue is always served in consecutive timeslots until it becomes empty. More formally, if $M_{ij}^{k-1} = 1$ and $M_{ij}^k = 0$, then it must be $r_{ij}^k = 0$. Otherwise, on the remaining input-output pairs, $\Omega(R)$ computes a maximal size matching. This is equivalent to the exhaustive service decomposition discussed in [28]. Since the complexity of a greedy maximal size matching is $O(N^2)$, then the overall computational complexity is $O(N^4)$.
- **GMin**: a *Gen-DEC* based algorithm. $\Omega(R)$ is a greedy minimum weight matching on R. Thus, the algorithm chooses the smallest elements in R, then it removes the corresponding row and column from R, and repeats the process until all the rows and columns in R are considered. Thus, the overall complexity is again $O(N^4 \log N)$.
- **Diag**: the matching selection is based on a precomputed set of N *covering diagonals* $D^k = [d_{ij}^k]$ on R, i.e. complete matchings with no elements in common and able to cover all the elements in R. Formally, $d_{ij}^k d_{ij}^h = 0$ for any $h \neq k$, and $\sum_{k=1}^N d_{ij}^k = 1$ for any i, j. As possibility, consider the set of matchings considered in the toy example of Sec. 2.3. The matching duration ϕ_k is chosen equal to the maximum value of the elements in the request matrix selected by D^k, i.e. $\phi_k = \max_{i,j}\{d_{ij}^k r_{ij} | r_{ij} > 0\}$ and the frame duration is

$\sum_{k=1}^{N} \phi_k$. The total number of iterations is N, each iteration with complexity $O(N)$ (since the maximum value among N elements of R must be found). Hence, the overall computational complexity is $O(N^2)$.

3.2. Frame Sorting

In this second phase of the frame definition algorithm, the matchings found in the frame $\mathcal{U}_R$ are now ordered to minimize the fatigue between consecutive timeslots. One simple way to model this problem is to consider an auxiliary graph. Each matching in $\mathcal{U}_R$ is associated with a vertex, and any pair of vertexes is connected by an edge, thus creating a complete graph by construction. The edge connecting the vertex M^k with M^h is tagged with the fatigue cost from one matching to the other, i.e. $E(M^k, M^h)$. By construction, we have that the cost of any path in the auxiliary graph corresponds to the fatigue cost needed to follow the particular sequence of matchings defined by the path. The frame sequence $\mathcal{F}_R^{\mathcal{P}}$ minimizing the fatigue can be computed from $\mathcal{U}_R$ by finding the minimum-cost Hamiltonian cycle, also known as the *Traveling Salesman Problem* (TSP), which is NP-complete. However, in our scenario, the edge costs satisfy the triangle inequality, and the problem reduces to a metric TSP [29], which is still NP-complete, but it can be simply approximated. We consider the following algorithms to sort $\mathcal{U}_R$, with the aim of approximating an average/best/worst case analysis:

- **No-Sort** (NS) does not modify the sequence of matchings.
- **Good-Sort** (BS) is a greedy algorithm that finds an approximated minimum cost cycle by visiting all vertexes: it chooses, at each step, the minimum cost edge towards an unvisited vertex. The initial vertex is chosen at random.
- **Bad-Sort** (WS) is a greedy algorithm that heuristically finds the maximum cost Hamiltonian cycle: starting from a random vertex, at each step, the maximum cost edge towards an unvisited vertex is chosen. This algorithm permits to define a worst-case sequence of matching that maximizes the fatigue, and it is useful to highlight the impact of the frame-sorting phase.

Notably, BS and WS do not provide the provably minimum and maximum cost cycle, respectively, thus the corresponding fatigue bounds could be loose.

The above frame-sorting algorithms can be freely combined with the matching-selection algorithms defined in the previous section. In the remainder of the paper, we use the notation (matching-selection)-(frame-sorting) to denote the particular pair of algorithms considered in our investigations: e.g. GMax-BS, GExa-NS, etc.

4. Theoretical Performance Analysis

We start to discuss the traffic scenarios adopted in the subsequent sections to compare the performance of the previously presented algorithms. We were able to evaluate *analytically* the performance of Diag, in Sec. 4.2, and GExa, Sec. 4.3.

4.1. Traffic Scenarios

The traffic exchanged among servers in a data center is a-priori unknown, since it depends on the actual applications running on the virtual machines, on their exchanged traffic and on the mapping between each virtual machine and the physical servers. In order to overcome such unpredictability and find general results, we devised a set of synthetic traffic scenarios aimed to test the algorithms performance under specific (benign or adversary) conditions.

We consider two main families of randomly generated, integer-value request matrices. In the case of hybrid data centers, we expect a non-negligible number of null elements in the request matrix, due to the fact that only elephant flows are routed across the OPT-fabric. Thus, we allow zero entries in the request matrix considered in the following cases.

The first family is denoted as *Average Sum* (AS), in which matrix elements r_{ij} are i.i.d. random variables, and satisfy the constraints: $E\left[\sum_{i=1}^{N} r_{ij}\right] = E\left[\sum_{j=1}^{N} r_{ij}\right] = \mu N$, i.e. the sum of each row and column is, on average, equal to a constant μN. Hence μ represents, given a flow, the average number of chunks present to each input at the beginning of the sampling period. Let $\textsc{Geom}(\mu)$ be a shifted geometric distribution with average μ, with $\mu \geq 1$. Among the family of AS request matrices, we consider:

- *Uniform* (UniAS): $r_{ij} = \textsc{Geom}(\mu)$. The coefficient of variation of the elements in R is $C_v = \sqrt{(\mu - 1)/\mu}$, which is always ≤ 1. Thus, the variance of the elements is relatively small.
- *Bidiagonal* (BidAS): let $M^1, M^2 \in \mathcal{M}$ be two randomly chosen permutation matrices. Set $r_{ij} = w_{ij}^1 m_{ij}^1 + w_{ij}^2 m_{ij}^2$ (with $0 < \alpha < 1$), where the $2N$ random weights (corresponding to non-null elements in M^1 and M^2) are chosen as $w_{ij}^1 = \textsc{Geom}(\alpha\mu N)$ and $w_{ij}^2 = \textsc{Geom}((1-\alpha)\mu N)$. Thus, R is obtained by summing two permutation matrices with random weights for each non-null element. On such family of matrices, any greedy approach is not able to find a maximum size matching, and for this reason this family can be considered "critical" during the matching selection phase.
- *Bimodal* (BimAS): $r_{ij} = 0$ with probability p and $\textsc{Geom}(\mu)$ with probability $1 - p$. Since the coefficient of variation is $C_v = \sqrt{\frac{(1+p)\mu - 1 + p}{\mu(1-p)}}$, we can set the values of p and μ to obtain a given C_v. For example, setting $p = 0.601$ and $\mu = 100$ gives $C_v \approx 2$. Note that, just for this scenario, the average sum of the rows and columns is $(1 - p)\mu N$. This family is similar to the uniform one, but with a larger variance.
- *Multidiagonal* (MudAS): let $\{M^k\}_{k=1}^{K}$ be a set of K permutation matrices: $M^k \in \mathcal{M}$. Now $r_{ij} = \sum_{k=1}^{K} w_{ij}^k m_{ij}^k$ (with $\alpha = \mu N/K$), where the KN random weights are chosen as $w_{ij}^k = \textsc{Geom}(\alpha)$. Thus, R is obtained by summing K permutation matrices with random weights and it is a generalization of BidAS matrix with K random matrices.

Table 2: Expected number of null and non-null elements for each request matrix

Request matrix	Non-null elements	Null elements
UniAS, UniPS	N^2	0
BidAS, BidPS	$2N$	$N^2 - 2N$
BimAS	$(1-p)N^2$	pN^2
MudAS,UMudAS, MudPS, UMudPS	KN	$N^2 - KN$

- *Unbalanced-Multidiagonal* (UMudAS): it is identical to MudAS but the weights of the K permutation matrices vary linearly: $r_{ij} = \sum_{k=1}^{K} w_{ij}^k m_{ij}^k$, where the KN weights are chosen as $w_{ij}^k = k\text{GEOM}(\alpha)$ and $\alpha \sum_{k=1}^{K} k = \mu N$.

We also consider the family of Perfect Sum (PS) matrices, whose rows and columns sum exactly to a constant μN: $\sum_{i=1}^{N} r_{ij} = \sum_{j=1}^{N} r_{ij} = \mu N$. Obviously, the elements $\{r_{ij}\}$ cannot be i.i.d.. PS matrices are an extension to the integer domain of double stochastic matrices, for which the BvN [26] decomposition was originally defined. Similarly to AS matrices, we consider the following PS families:

- *Uniform* (UniPS): choose a set of μN random permutation matrices $M^k \in \mathcal{M}$ and compute $R = \sum_{k=1}^{\mu N} M^k$. UniPS matrices are characterized by elements with low variance, because, for the Central Limit Theorem (CLT), $C_v \to 0$, as $N \to \infty$.

- *Bidiagonal* (BidPS): let $M^1, M^2 \in \mathcal{M}$ be two random matching matrices: $R = \alpha\mu N M^1 + (1-\alpha)\mu N M^2$, with $0 < \alpha < 1$.

- *Multidiagonal* (MudPS): as generalization of BidPS, a set of K random permutation matrices $M^k \in \mathcal{M}$ is chosen and combined as $R = \sum_{k=1}^{K} \alpha M^k$, where $\alpha = \mu N/K$.

- *Unbalanced Multidiagonal* (UMudPS): it is a variant of MudPS, where the weight of each permutation matrix varies linearly: $R = \sum_{k=1}^{K} \alpha k M^k$, where $\alpha = 2\mu N(K+1)/K$.

For an easy reference, Table 2 reports the expected number of non-null elements in the request matrices considered in our work.

4.2. Theoretical Performance of Diag algorithm

The fatigue cost and throughput performance of Diag algorithm can be evaluated analytically for some of the above scenarios.

For the constant uniform request matrix of the toy-example of Sec. 2.3, Diag computes frame $\mathcal{F}_2$ of Table 1 and thus behaves optimally in terms of both maximum sustainable load and fatigue cost. Whenever the request matrix is different from constant uniform, then its performance can be quite poor, as shown in the following.

We start now to evaluate the reconfigurations for a generic request matrix. In the case of Diag, they can be evaluated easily because the variation between matchings is always equal to 2 for each non-null r_{ij}. Hence, for a frame of k distinct matchings, with $k \le N$, the overall reconfigurations are always upper bounded by $2kN$. Note that this holds independently from the matching sorting phase, since the final frame $\mathcal{F}_R^{\mathcal{P}}$ is independent from it. Table 3 provides the average fatigue cost per chunk for all considered traffic scenarios, assuming $r_{ij} > 0$ for any i, j. For UniAS and BidAS that value represents an upper bound on the actual fatigue costs (due to possibly zero values in R), for BimAS it is an average, whereas for UniPS and BidPS this fatigue cost is exact. In Sec. 4.3 we will show that the results in Table 3 hold also for GExa.

Table 3: Fatigue costs for Diag and GExa algorithms

	Request matrix		
	UniAS UniPS	BidAS BidPS	BimAS
Total reconfigurations	$2N^2$	$4N$	$2(1-p)N^2$
Ave. total num. of chunks	μN^2	μN^2	$\mu(1-p)N^2$
Ave. fatigue per chunk	$2/\mu$	$4/(\mu N)$	$2/\mu$

After evaluating the fatigue cost of Diag, we now move to the throughput. To evaluate the performance of Diag algorithm under different request matrices, we refer to the results obtained in the appendix using extreme value theory [30]. The following three theorems refer to the three different traffic scenarios: UniAS, UniPS and BimPS.

Theorem 1. *Let $R = [r_{ij}]$ be a UniAS request matrix, with $E[r_{ij}] = \mu \gg 0$. R is sustainable under Diag algorithm with the expected value of the frame-expansion factor S that can be upper bounded as follows:*

$$E[S] \le \frac{\log N + \gamma}{\left(1 + \frac{\Gamma(N)}{\sqrt{N}}\right)}$$

Proof. To evaluate $E[S]$, we start to compute the average value of $F_R^{\mathcal{P}}$ and then we evaluate the average value of T_R. Let us focus on $F_R^{\mathcal{P}}$. Let C_d be the maximum element along the d-th diagonal of R. By construction, under Diag policy, $F_R^{\mathcal{P}} = \sum_{d=1}^{N} C_d$. We now wish to evaluate the average frame size $E[F_R^{\mathcal{P}}]$. $C_1, C_2, \ldots, C_N$ are i.i.d. random variables. Then

$$E[F_R^{\mathcal{P}}] = NE[C_d] \tag{4}$$

$C_d = \max_{i=1\ldots N}\{A_i\}$, i.e. the maximum of N i.i.d. random variables A_i, distributed as each element of R. Since r_{ij} is geometrically distributed with average $\mu \gg 0$, we can approximate A_i with an exponential distribution with average μ. By Lemma 2 in appendix and (4),

$$E[F_R^{\mathcal{P}}] = \mu N(\log N + \gamma) \tag{5}$$

Let us now focus on T_R. Define T'_R and T''_R as the maximum row and column sums of R, i.e. $T'_R = \max_{j=1,\ldots,N} \sum_{i=1}^{N} r_{ij}$,

$T''_R = \max_{i=1,\dots,N} \sum_{j=1}^{N} r_{ij}$ From the Birkhoff-von Neumann theorem [26], $T_R = \max\{T'_R, T''_R\}$. Since all r_{ij} are i.i.d., we can focus on a generic row i of R and evaluate the sum B_i of the corresponding values: $B_i = \sum_{j=1}^{N} r_{ij}$. Thanks to the Central Limit Theorem (CLT), the distribution of B_i tends to the Normal distribution

$$B_i \sim \mathcal{N}(\mu N, \mu^2 N) \tag{6}$$

Rewriting T'_R as $T'_R = \max_{i=1,\dots,N}\{B_i\}$, from Lemma 3 in appendix

$$E[T'_R] \to \mu N + \mu\sqrt{N}\Gamma(N) \tag{7}$$

where $\Gamma(N)$ is a function defined in (A.4) of the appendix and grows as $\Gamma(N) \approx \sqrt{2\log N}$ for $N \to \infty$. Since $T_R \geq T'_R$ (stochastically), the right side of (7) represents a lower bound on $E[T_R]$. Combining (5) and (7), the frame-expansion ratio S is upper bounded by: $E[S] \leq (N(\log N + \gamma))/(N + \sqrt{N}\Gamma(N))$, which corresponds to our main claim. □

Theorem 2. *Let $R = [r_{ij}]$ be a UniPS request matrix, being $E[r_{ij}] = \mu$. R is sustainable under Diag algorithm with a frame-expansion factor S whose average is:*

$$E[S] = 1 + \sqrt{\frac{1}{\mu}\left(1 - \frac{1}{N}\right)}\Gamma(N)$$

Proof. By construction, $T_R = \mu N$. We need to evaluate $E[F_R^{\mathcal{P}}]$ to compute $E[S]$. All the elements r_{ij} of the request matrix are identically distributed, even if not independent. Say A is the random variable corresponding to any r_{ij}. Now A is obtained by summing μN complete matchings, each of them including the element (i, j) with probability $1/N$. This is equivalent to state that $A = \sum_{i=1}^{\mu N} H_i$ with $H_i = 1$ with prob. $1/N$ and $H_i = 0$ with prob. $1 - 1/N$. Thanks to the CLT, A is normally distributed: $A \sim \mathcal{N}(\mu, \mu(1 - 1/N))$. Define C as the maximum along a particular diagonal; C is the maximum of N i.i.d. random variables distributed as A. By Lemma 3 in appendix, $E[C] \to \mu + \sqrt{\mu(1 - 1/N)}\Gamma(N)$. Since $E[F_R^{\mathcal{P}}] = NE[C]$, we obtain our final assert: $E[S] = (N\mu + N\sqrt{\mu(1 - 1/N)}\Gamma(N))/(N\mu)$. □

Theorem 3. *Let $R = [r_{ij}]$ be a Bim-AS request matrix, being $E[r_{ij}] = \mu(1 - p)$. R is sustainable under Diag algorithm with a frame-expansion factor S whose average can be upper bounded as:*

$$E[S] \leq \frac{\log(1 - p) + \log(N) + \gamma}{1 - p + \frac{\Gamma(N)}{\sqrt{N}}\sqrt{1 - p^2}} \tag{8}$$

Proof. We can repeat the same arguments as the proof of Theorem 1. To compute $E[F_R^{\mathcal{P}}]$, thanks to Lemma 1 in appendix, simply substitute $\log N$ with $\log N + \log(1 - p)$. For $E[T'_R]$, it can be shown that B_i is normally distributed as $\mathcal{N}(\mu N(1 - p), N\mu^2(1 - p^2))$. The result immediately follows. □

Fig. 5 shows the average frame-expansion ratio obtained by simulating a large number of request matrices for different values N. We have investigated the different families of request matrices: UniAS, BimAS with $C_v = 2$ (large variance) and $C_v = 4$ (very large variance) and UniPS. The points (SIM) refer to the results obtained by simulation, and the curves (TEO) refer to the analytical curves of Theorems 1, 2 and 3. The graphs show that the bounds of Theorems 1 and 3 are quite tight, especially for large N, and the approximation of Theorem 2 is very accurate.

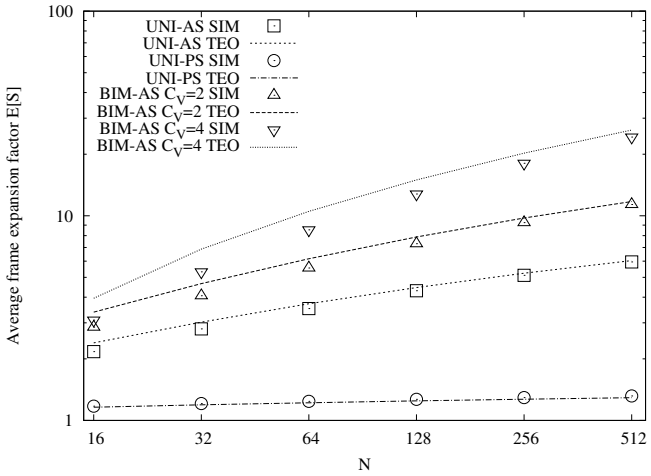

Figure 5: Analytical and simulated results for the average frame-expansion ratio $E[S]$ and for different request matrices under the Diag algorithm.

Table 4: Approximated performance for Diag and GExa when $N \to \infty$

Algorithm	Request matrix	Maximum sustainable load
Diag	UniAS, BimAS	$\geq \sqrt{\frac{2}{N \log N}}$
	UniPS	$\geq \sqrt{\frac{1}{2 \log N}}$
GExa	any	≥ 0.5

We can assess the maximum sustainable load by recalling that it grows as 1/S, as discussed at the end of Sec. 2.1. Table 4 shows the approximated performance of Diag for enough large N, evaluated by the computing the limit for $N \to \infty$ for the relations proved in the previous theorems. For both cases, the throughput, in the worst case, decreases quite fast as N increases. Recall, that this low performance is traded with minimum computational complexity and minimum fatigue.

4.3. Theoretical Performance of GExa algorithm

We now evaluate the fatigue cost and the throughput for GExa algorithm. Since the service is exhaustive for each input-output pair, the reconfigurations are always 2 for each non-null r_{ij}, as for Diag. Hence, Table 3 is also valid for GExa and we can claim that GExa is optimal in terms of fatigue and thus fabric lifetime. Regarding the performance, we can claim:

Theorem 4. *Let $R = [r_{ij}]$ be any request matrix. R is sustainable under the GExa algorithm with a frame-expansion factor $S \leq 2$.*

Proof. Observe that GExa decomposes R using a sequence of maximal matchings. From Theorem 2.2 in [25] or Theorem 4.2

Table 5: Average fatigue cost per chunk for UniAS request matrices when μ = 100, for a 64×64 switch

Decomposition algorithm	Good Sort (WS)	No Sort (NS)	Bad Sort (BS)
BvN	0.872	0.810	0.741
GMax	0.425	0.415	0.377
GMin	0.723	0.045	0.049
GExa	0.866	0.020	0.020
Diag	0.020	0.020	0.020

in [28], if a matrix is decomposed by *any* sequence of maximal matchings, then the number of matchings needed is at most twice than the number obtained by BvN. Hence, $F_R^{\mathcal{P}} \leq 2T_R$ and thus $S \leq 2$. □

Note that this result is generic since holds for any request matrix. As a consequence, the maximum sustainable load by GExa is at least 0.5, independently of the switch size. By comparing GExa and Diag in Tab. 4, GExa is largely outperforming Diag in terms of performance, but with the same fatigue costs. As a conclusion, GExa must be preferred to Diag as matching selection algorithm.

5. Simulative performance analysis

In this section we evaluate the fatigue costs and the performance of all the combinations of matching selection and sorting algorithms, through Montecarlo simulations. In this way, we were able to extend the analysis to a wider scenarios than the ones considered in the previous sections and chosen for being amenable to analytical treatment.

Results have been obtained through an ad-hoc simulator in C language.

The average number of chunks per input flow, μ, is set equal to 100. All simulations' results are obtained as an average of 100 simulation runs, each one with a different randomly generated request matrix, to obtain statistically significant results. In all the reported results, we have evaluated the 95% confidence interval, using the batch mean approach; we have observed always a relative error varying between 0.1% and 3% at most.

In our results, we will evaluate the frame-expansion ratio and the fatigue cost. We do not report the achievable delays, which are discussed in [7] and depend on the particular arrival process and all the temporal latencies typical of the adopted implementation.

5.1. Effect of Frame Sorting

We first evaluate the effect of the algorithms to sort the frame. Table 5 reports the fatigue cost by combining a specific matching selection algorithm with a particular sorting algorithm, under UniAS request matrices in a 64×64 OPT-ToR switch. Very similar results were obtained for different switch sizes and different random request matrices. The Diag algorithm is not affected by the sorting and the fatigue cost per chunk is coherent with the analytical formula $2/\mu$ reported in Table 3. Recall that $2/\mu$ is the minimum cost achievable by any algorithm under the UniAS scenario, but it requires a large frame-expansion factor S, as shown in Theorem 1.

As a general comment for the results of Table 5, the beneficial effect of the frame-sorting algorithm on the fabric lifetime depends from the specific matching-selection algorithm. By construction, in general, we could expect that Good-Sort (BS) will outperform No-Sort (NS) which, in turn, will outperform Bad-Sorting (WS). This is not always true, as discussed below.

For the BvN matching-selection algorithm, BS allows to reduce the fatigue costs by 10% with respect to NS, and 17% with respect to WS. In all cases, BvN shows the largest number of reconfigurations, and this is due to the specific algorithm adopted in BvN, based on computing a maximum size matching at each iteration, without considering the cost to change the matching.

When combined with GMax, BS reduces the fatigue cost similarly for the BvN case. In absolute terms, the reconfigurations are less than BvN, since the greedy algorithm based on the queue length induces a correlation between the matchings computed in subsequent iterations of the algorithm.

The effect of the correlation is highlighted in GMin, where the frame-sorting has always a negative effect. Indeed, fatigue costs without sorting (i.e. with NS) are already very small, and increases with BS. This is not surprising, since BS is an approximated algorithm to solve the TSP problem and its solution is worse than the initial sequence offered by keeping $\mathcal{U}_R$ unsorted. Although not completely intuitive at a first glance, this effect is due to the particular metric used to compute the matching at each iteration. By subtracting the minimum weight matching M^k from R^k at iteration k, there is a high probability that the new minimum weight matching M^{k+1} shares some (at most, $N-1$) edge with M^k. This correlation induces an efficient "self-sorting" property, providing an efficiency comparable with, and in some situations even better, than the one achieved by BS sorting. On the contrary, when running GMax algorithm, M^k is a (almost) maximum weight matching; as such, there is a very low probability that M^{k+1} shares edges with M^k. This explain the fewer reconfigurations of GMin with respect to GMax. As a remark, GMin is the only matching selection algorithm which appear to be efficient in terms of lifetime without any additional sorting algorithm.

Similarly to Diag, GExa-NS and GExa-BS are both optimal in terms of fatigue, since the matching order induced by GExa is already optimal. On the contrary, WS changes the order and (differently from Diag), the fatigue increases.

Since the above reported results hold qualitatively in many scenarios, we focus only on the following optimized combinations of frame scheduling algorithms in the next sections: BvN-BS, GMax-BS, GMin-NS, Diag-NS and GExa-NS. These algorithms have very different computational complexities and memory requirements; the sorting procedure itself requires to store the whole frame sequence to sort it. The ranking among the algorithms in terms of increasing complexity is: Diag-NS (less complex), GExa-NS, GMin-NS, GMax-BS and BvN-BS (more complex).

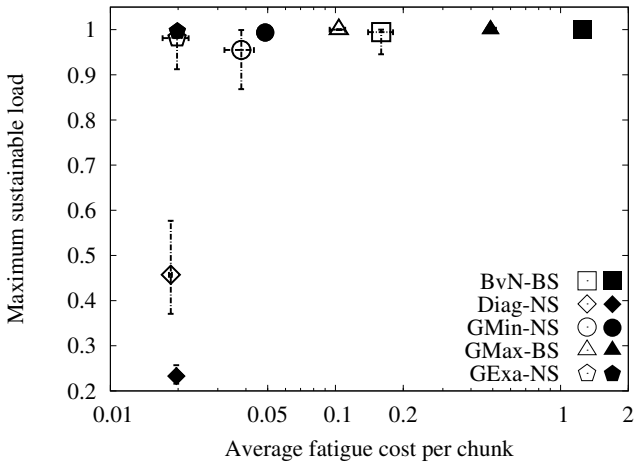

Figure 6: Throughput and fatigue tradeoff under UniAS traffic for $N = 16$ (white shapes) and $N = 128$ (black shapes).

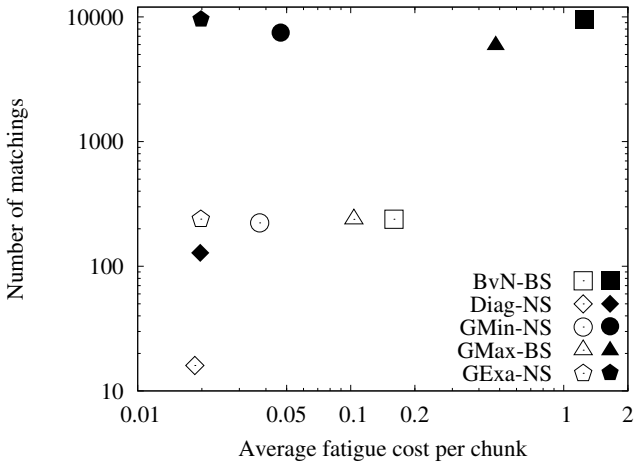

Figure 7: Average number of matchings per frame and fatigue tradeoff under UniAS traffic for $N = 16$ (white shapes) and $N = 128$ (black shapes).

5.2. Switching Fatigue and Throughput Tradeoff

We mainly report the results for $N = 16$ (denoted with white shapes in the graphs) and $N = 128$ (denoted with black shapes in the graphs); however, similar results hold also for $N = 32$ and $N = 64$, and are not reported here for the sake of conciness.

In all the reported plots, each point corresponds to the average value; two bars around each point (one horizontal bar and one vertical bar) show the maximum and minimum values obtained considering all 100 runs. When the error-bars are not visible, the results of each run are almost identical to the average value, i.e., results do not change for different seeds to generate the random matrices. The traffic matrices MudAS, UMudAS, MudPS, UMudPS have been obtained by fixing $K = 10$.

Fig. 6 shows the tradeoff between the maximum sustainable load (introduced in Sec. 2.1) and average fatigue cost per chunk obtained by the different algorithms, under UniAS traffic. This kind of graphs provides a direct view of the tradeoff between performance and foreseen lifetime of the fabric. Indeed, it is possible to compare both the achievable maximum load and the fatigue between different frame scheduling algorithms. We provide as reference also Fig. 7, that shows the number of distinct matchings computed by each algorithm.

All the algorithms achieve almost the maximum throughput, except for Diag-NS whose maximum sustainable load decreases with N as shown in Theorem 1 and in Table 4. GExa-NS is optimal from the throughput point of view, and in this case the bound provided by Theorem 4 is loose since actually $S \approx 1$. As expected, Diag-NS and GExa-NS achieve the minimum fatigue, coherently with the formulas in Table 3. Fig. 7 shows an interesting difference between the two: while Diag-NS uses exactly N matchings (i.e. the minimum possible), GExa-NS uses almost the largest number of distinct matchings in a frame. Indeed, roughly $N^2\mu$ chunks must scheduled in a frame, and, to achieve the maximum frame load, each single matching should roughly serve N chunks. Hence, there are at most $N\mu$ different matchings in a generic frame under UniAS and GExa-NS adopts a number of distinct matchings comparable with the maximum allowed one. Nevertheless, in GExa-NS the overall fatigue is small because the variations between any pair of consecutive matchings are very small.

GMin-NS, despite its relative simplicity, offer a reasonable tradeoff, since achieves almost the maximum throughput with a fatigue only 2-3 times larger than Diag-NS and GExa-NS. In Fig. 8 we focus on the tradeoff obtained by GExa-NS and GMin-NS by varying N under UniAS traffic. Regardless of fabric size, the maximum sustainable load is always significant for both algorithms, and the growth of the fatigue as a function of N is marginal for GMin-NS and null for GExa-NS.

Coming back to Fig. 6, both BvN-BS and GMax-BS achieve the maximum throughput, but, as observed in Sec. 5.1, GMax-BS shows lower fatigue costs than BvN-BS, due to the metrics used to compute the matching. In particular, the fatigue in BvN-BS is between 6 and 50 times larger than GExa-NS, and for GMax-BS is between 5 and 25 times. Finally, fatigue increases for larger switch size, as expected. In general, GExa-NS reveals the best overall tradeoff, with minimum fatigue and almost maximum throughput.

Fig. 9 shows the performance of the previous algorithms under BimAS scenario. The same observations as in the previous UniAS scenario holds, also from a quantitatively point of view.

In the case of UniPS scenario, Fig. 10 shows that BvN-BS is the worst algorithm in terms of fatigue, especially when the switch size grows. The best results are obtained by GExa-NS and Diag-NS, the latter providing higher throughput, differently from the UniAS scenario. This is mainly due to the smaller variance of the values in the diagonal elements of the request matrix R: the maximum element on a diagonal is close to the average and Diag-NS shows better performance. GExa-NS and GMin-NS show the best tradeoff, since they achieve the minimum fatigue, close to Diag-NS, and almost the maximum throughput.

We now consider the BidPS scenario. Because of the particular traffic matrix, the average fatigue cost is always equal to $4/\mu/N$, but different algorithms achieve different throughput, as shown in Table 6. Diag-NS is the worst and achieves

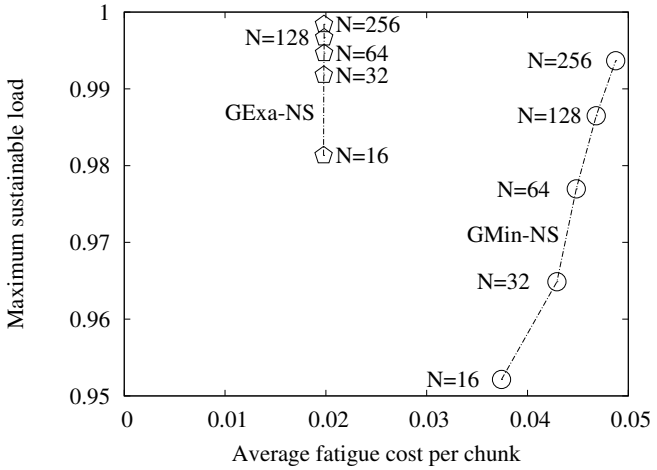

Figure 8: Throughput and fatigue tradeoff for GMin-NS and GExa-NS under UniAS traffic.

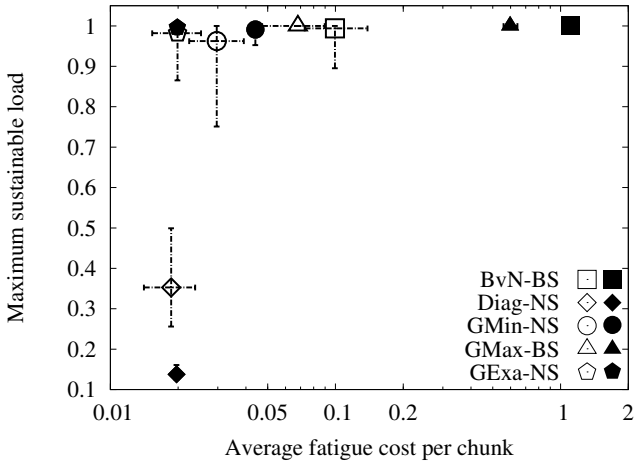

Figure 9: Throughput and fatigue tradeoff under BimAS traffic for $N = 16$ (white shapes) and $N = 128$ (black shapes).

Table 6: Performance under BidPS scenario with $\alpha = 1/3$

Algorithm	Maximum sustainable load		Average fatigue cost per chunk	
	$N = 16$	$N = 128$	$N = 16$	$N = 128$
BvN-BS	1.00	1.00	0.0025	0.00031
GMax-BS	1.00	1.00	0.0025	0.00031
GMin-NS	1.00	1.00	0.0025	0.00031
GExa-NS	0.75	0.75	0.0025	0.00031
Diag-NS	0.12	0.02	0.0025	0.00031

a throughput around $2/N$ since it requires N slots to serve one packet from each of the two non-empty queues. Because of its poor performances, we will omit Diag-NS from the following results. GExa-NS suffers a 25% throughput reduction, and this is due to the greedy approach adopted in the decomposition. Table 7 shows the performance under MudPS traffic, which is

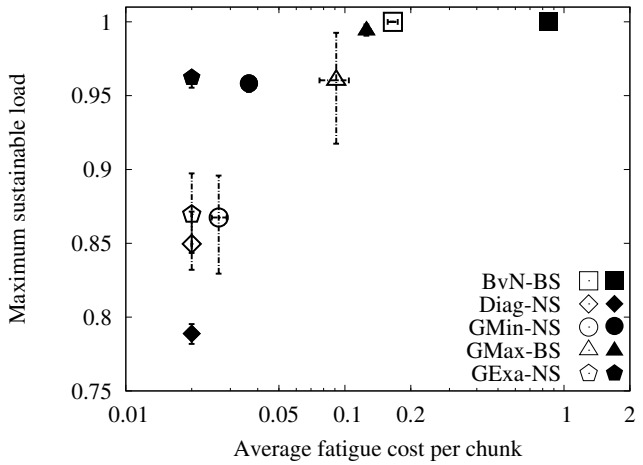

Figure 10: Throughput and fatigue tradeoff under UniPS scenario for $N = 16$ (white shapes) and $N = 128$ (black shapes).

Table 7: Performance under MudPS scenario ($K = 10$)

Algorithm	Maximum sustainable load		Average fatigue cost per chunk	
	$N = 16$	$N = 128$	$N = 16$	$N = 128$
BvN-BS	1.00	1.00	0.0109	0.0015
GMax-BS	0.75	0.64	0.0113	0.0015
GMin-NS	0.75	0.64	0.0099	0.0015
GExa-NS	0.72	0.63	0.0095	0.0015

a generalization of BidPS scenario with $K = 10$ permutation matrices. From the fatigue point of view, all the algorithms behave the same as in BidPS, but now all the greedy approaches for the frame decomposition obtain a throughput lower than the throughput-optimal BvN. This is expected, since BidPS and MudPS are carefully crafted to avoid maximum throughput for any greedy approach.

In Table 8 we show the performance under UMudPS traffic, built with unbalanced permutation matrices. Also this scenario shows some throughput impairment for all greedy approaches, but now the fatigue cost of GExa-NS appear to be around 4 times smaller than BvN-BS.

Finally, Table 9 reports the performance achieved by BimAS. In terms of throughput, all the algorithms achieve almost

Table 8: Performance under UMudPS scenario ($K = 10$)

Algorithm	Maximum sustainable load		Average fatigue cost per chunk	
	$N = 16$	$N = 128$	$N = 16$	$N = 128$
BvN-BS	1.00	1.00	0.0308	0.0061
GMax-BS	0.85	0.86	0.0201	0.0043
GMin-NS	0.77	0.80	0.0111	0.0014
GExa-NS	0.81	0.76	0.0086	0.0014

Table 9: Performance under BidAS scenario, with $\alpha = 1/3$

Algorithm	Maximum sustainable load		Average fatigue cost per chunk	
	$N = 16$	$N = 128$	$N = 16$	$N = 128$
BvN-BS	1.00	1.00	0.0060	0.0043
GMax-BS	1.00	1.00	0.0040	0.0014
GMin-NS	0.98	0.99	0.0029	0.0004
GExa-NS	0.99	0.99	0.0025	0.003

Table 10: Performance under MudAS scenario ($K = 10$)

Algorithm	Maximum sustainable load		Average fatigue cost per chunk	
	$N = 16$	$N = 128$	$N = 16$	$N = 128$
BvN-BS	1.00	1.00	0.0583	0.0891
GMax-BS	1.00	1.00	0.0407	0.0579
GMin-NS	0.93	0.94	0.0157	0.0027
GExa-NS	0.97	0.98	0.0093	0.0015

the maximum throughput. Instead, among such throughput-optimal policies, only GExa-NS is able to obtain an average fatigue cost equal to the minimum one, corresponding to $4/\mu/N$ already computed in the BimPS scenario. Thus, the throughput degradation of GExa-NS observed in BimPS appears to depend on the deterministic values of the traffic matrix; some random perturbation in the BimPS matrix, resulting into BimAS, allows GExa-NS to achieve the maximum throughput, while keeping an optimal behavior in terms of fatigue costs, differently from all other algorithms.

In conclusion, GExa-NS offers the best tradeoff in terms of throughput, fatigue costs and computational complexity, since in most of the scenarios it obtains the maximum throughput while keeping a minimum fatigue cost.

5.3. Real traffic matrices

To further investigate the behavior of the proposed algorithms, we evaluated their performance under real case scenarios, in particular using the data demand measured in two production data centers, respectively from Facebook [9] and Microsoft [31].

Table 11: Performance under UMudAS scenario ($K = 10$)

Algorithm	Maximum sustainable load		Average fatigue cost per chunk	
	$N = 16$	$N = 128$	$N = 16$	$N = 128$
BvN-BS	0.99	0.99	0.0533	0.0820
GMax-BS	1.00	1.00	0.0331	0.0465
GMin-NS	0.93	0.94	0.0141	0.0025
GExa-NS	0.97	0.98	0.0085	0.0014

Table 12: Performance under a real traffic matrix measured in a Facebook data center

Algorithm	Maximum sustainable load	Fatigue cost per chunk
BvN-BS	0.99	0.578
GMax-BS	1.00	0.111
GMin-NS	0.92	0.055
GExa-NS	0.93	0.044
Diag-NS	0.79	0.015

In [9] authors collected 24-hours traces from a Facebook's data center, gathering different classes of traffic matrices. The measurements refer to the communication among 64 ToRs, corresponding to a 64×64 traffic matrix. Around 80% of the traffic is generated within the data center (denoted usually as east-west traffic). Two traffic matrices are claimed to be typical in Facebook data centers: either the traffic is homogenous for Hadoop/MapReduce applications or it is very unbalanced. From the numerical values shown in [9], the first case is very similar to the constant uniform request matrix considered in the toy example of Sec. 2.3, for which all the algorithms (also Diag-BS) behave exactly the same and obtain the maximum sustainable load and the minimum fatigue costs, corresponding to frame $\mathcal{F}_2$ in Table 1. The second scenario with unbalanced traffic is due to the traffic from/to the front-end servers of the main Facebook services, running web-services and caching applications. The traffic is strongly unbalanced, with 75%of ToR switches sending most of their traffic toward 25% of destination ToRs and vice versa.

In our investigation, we exactly adopted the unbalanced traffic matrix shown in Fig. 5(b) of [9] and showed the results in Table 12 for the most relevant algorithms considered so far. Note that the Facebook traffic matrices have been reported in relative values and thus we had to renormalize them coherently with the previous synthetic traffic matrices. From our results, all considered algorithms are able to reach at least 90% of maximum sustainable load, except for Diag-NS which obtains a slightly lower load but with the minimum fatigues cost (equal to $1/64$). These results corroborate qualitatively the results observed in the previous traffic scenarios. In more details, GExa-NS outperforms GMax-BS in term of fatigue by a factor 3.

We also considered the performance achievable for another traffic matrix, that was measured by the authors of [31] in a Microsoft data center, composed by 73 ToR switches connecting around 1500 servers. The main applications running in such data center were data mining based on Map Reduce and other distributed storage applications. The measured traffic matrix appears to be unbalanced (similarly to Facebook data center) but now it is also quite sparse with a few set of hot-spot ToR switches that send and receive most of the traffic. Table 13 shows the performance under the traffic matrix reported in Fig. 2 of [31]. All the algorithms achieve the maximum throughput, thanks to the sparsity of the traffic matrix which appears "simple to schedule". The only exception is Diag-NS,

Table 13: Performance under a real traffic matrix measured in a Microsoft data center

Algorithm	Maximum sustainable load	Fatigue cost per chunk
BvN-BS	1.00	1.273
GMax-BS	1.00	1.247
GMin-NS	1.00	0.224
GExa-NS	1.00	0.152
Diag-NS	0.18	0.013

which is completely inefficient in terms of throughput due to sparsity of the traffic matrix and thus cannot be considered a practical solution for scheduling. Instead, the behavior in term of fatigue cost is quite different among the various algorithms, due to the large variance in the value of the traffic matrix. BvN-BS and GMax-BS show a similar behavior with respect to UniAS scenario. Instead, excluding Diag-NS, GExa-NS is the best algorithm, reducing the fatigue cost by a factor 10 with respect to BvN-BS and GMax-BS.

Thus, under traffic matrices typical in Facebook and Microsoft data centers, GExa-NS offers the best tradeoff between complexity, throughput and fatigue.

6. Related Work

Proactive traffic scheduling has been proposed [10, 15, 16] to switch traffic based on a-priori knowledge of the traffic demands between racks or ToR switches. All these works are completely oblivious of the particular nature of the optical data center fabric, assuming an ideal non-blocking behavior without reconfiguration delays and with unlimited lifetime. Furthermore, they assume to know the traffic matrix in advance. The proposed approaches are always based on a standard Birchkoff-von Neumann decomposition, which is also considered in our paper as term of comparison (see Sec. 3.1). We have shown that such approach applied in our scenario maximizes throughput but not the OPT-fabric lifetime.

In optical switches based on MEMS, tunable lasers and other technologies, a reconfiguration latency must be paid when the switching fabric changes configuration; a "blackout" period is experienced in packet transmissions, during which the whole switching fabric is not available to transfer packets. In [32] the optimal frame scheduling to compute was studied in such scenario. The cost function minimized in [32] is similar to the one considered in our paper since it considers a reconfiguration cost paid anytime the switching configuration changes. However, differently from our case, the cost in [32] is independent of the number of input-output connections that change inside the switching fabric: a single connection modification implies that the whole switching fabric becomes unavailable, thus introducing the cost of a complete reconfiguration. Hence, differently from ours, the scheduling policy is designed to minimize the number of matchings to serve all the packets in the request matrix and not the number of variations in the matchings. Notably, [32] showed that the optimal scheduling problem with reconfiguration latency belongs to the NP-complete class, and proposed two sub-optimal algorithms Min and Double. Min algorithm decomposes the request matrix into N matchings as our Diag, but the corresponding frame-expansion factor S grows as $S \approx 4 \log_2 N$. As an alternative, Double algorithm decomposes the request matrix with $2N$ matchings while keeping $S = 2$.

Regarding the design of hybrid data centers, an alternative implementation with respect to [7], often referred in our work, is proposed in [12], which adopts a 320 port MEMS switch as OPT-fabric.

The authors propose VLAN tagging based on OpenFlow (at the switch) and OpenvSwitch (at the server) to forward the traffic flows into the OCS network. Notably, their proposed implementation is compatible with the scheduling approach proposed in our work. But, differently from our work, [12] supports multi-hop routing, according to which the final ToR switch is reached through a sequence of intermediate ToR switches connected to the OPT-fabric.

An alternative hybrid data center architecture has been proposed in [18], built with a MEMS-based OCS network interconnecting the ToR switches as in our work. The frequency at which the scheduler computes the solution adapts to the traffic conditions and it is not fixed as in our case. The main scheduling algorithm is based on the maximum weight matching computed on the number of chunks enqueued in the ToR queues; this approach is very similar to GMax considered later in our work.

In conclusion, all the previous implementation and design efforts have not considered the limited lifetime of a MEMS-based OCS-fabric. Our traffic scheduling approach can be adopted in all these cases.

7. Conclusions

We have considered an optical fabric to interconnects ToR switches in a data center, allowing fast reconfigurable circuit switching among the racks. We have assumed that MEMS are adopted for full optical switching, and this technology is affected by limited lifetime due to the mechanical fatigue.

We addressed the problem of scheduling the traffic across racks to maximize the performance in terms of throughput and to maximize the lifetime of the switching fabric. The main idea is to minimize the number of single variations between consecutive switching configurations. We have proposed a family of fatigue-aware frame scheduling algorithms that offer different tradeoff between performance (throughput and fatigue) and computational complexity. We have investigated the achieved tradeoffs using both analytical and simulative approaches, under both synthetic and realistic traffic matrices observed in operational Facebook and Microsoft data centers. As result, we have shown that, in a majority of the scenarios, the specific algorithm denoted as GExa-NS, is outperforming all the other approaches based on state-of-art algorithms. Notably, GExa-NS can substitute the standard decomposition algorithms for

the traffic scheduling in optical fabrics, without incurring in any extra computational cost, but providing an important improvement in the lifetime of the MEMS-based switches.

References

[1] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. E. Ng, M. Kozuch, M. Ryan, c-Through: Part-time optics in data centers, SIGCOMM Computer Communication Review 40 (4) (2010) 327–338.

[2] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, A. Vahdat, Helios: A hybrid electrical/optical switch architecture for modular data centers, ACM SIGCOMM, ACM, New York, NY, USA, 2010, pp. 339–350.

[3] H. Wang, Y. Xia, K. Bergman, T. E. Ng, S. Sahu, K. Sripanidkulchai, Rethinking the physical layer of data center networks of the next decade: Using optics to enable efficient *-cast connectivity, ACM SIGCOMM Computer Communication Review 43 (3) (2013) 52–58.

[4] M. Chen, H. Jin, Y. Wen, V. Leung, Enabling technologies for future data center networking: a primer, IEEE Network 27 (4) (2013) 8–15.

[5] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, A. Tanwer, FireFly: A reconfigurable wireless data center fabric using free-space optics, ACM SIGCOMM, ACM, New York, NY, USA, 2014, pp. 319–330.

[6] A. Hammadi, L. Mhamdi, A survey on architectures and energy efficiency in data center networks, Computer Communications, Elsevier 40 (2014) 1–21.

[7] H. Liu, F. Lu, A. Forencich, R. Kapoor, M. Tewari, G. M. Voelker, G. Papen, A. C. Snoeren, G. Porter, Circuit switching under the radar with REACToR, in: ACM/USENIX NSDI, 2014.

[8] T. Benson, A. Anand, A. Akella, M. Zhang, Microte: Fine grained traffic engineering for data centers, in: CoNEXT, ACM, New York, NY, USA, 2011, pp. 8:1–8:12.

[9] A. Roy, H. Zeng, J. Bagga, G. Porter, A. C. Snoeren, Inside the social network's (datacenter) network, in: ACM SIGCOMM, ACM, New York, NY, USA, 2015, pp. 123–137.

[10] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, A. Vahdat, Integrating microsecond circuit switching into the data center, in: ACM SIGCOMM, ACM, New York, NY, USA, 2013, pp. 447–458.

[11] Big data fabric (Oct. 2015).
URL `http://www.plexxi.com`

[12] K. Christodoulopoulos, D. Lugones, K. Katrinis, M. Ruffini, D. O'Mahony, Performance evaluation of a hybrid optical/electrical interconnect, IEEE/OSA Journal of Optical Communications and Networking 7 (3) (2015) 193–204.

[13] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, Y. Chen, OSA: An optical switching architecture for data center networks with unprecedented flexibility, Networking, IEEE/ACM Transactions on 22 (2) (2014) 498–511.

[14] K. Chen, X. Wen, X. Ma, Y. Chen, Y. Xia, C. Hu, Q. Dong, Wavecube: A scalable, fault-tolerant, high-performance optical data center architecture, in: IEEE INFOCOM, 2015, pp. 1903–1911.

[15] B. C. Vattikonda, G. Porter, A. Vahdat, A. C. Snoeren, Practical TDMA for datacenter Ethernet, in: EuroSys, 2012.

[16] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, H. Fugal, Fastpass: A centralized "zero-queue" datacenter network, in: ACM SIGCOMM, 2014.

[17] S series optical circuit switch (Oct. 2015).
URL `http://www.calient.net`

[18] C.-H. Wang, T. Javidi, G. Porter, End-to-end scheduling for all-optical data centers, in: IEEE INFOCOM, 2015. doi:10.1109/INFOCOM.2015.7218406.

[19] T.-W. Yeow, K. Law, A. Goldenberg, MEMS optical switches, IEEE Communications Magazine 39 (11) (2001) 158–163.

[20] M. C. Wu, O. Solgaard, J. E. Ford, Optical MEMS for lightwave communication, Journal of Lightwave Technology 24 (12) (2006) 4433–4454.

[21] MEMS 4×4 switch (Oct. 2015).
URL `http://www.agiltron.com`

[22] J. Walraven, Failure mechanisms in mems, in: Test Conference, 2003. Proceedings. ITC 2003. International, Vol. 1, 2003, pp. 828–833.

[23] W. M. van Spengen, {MEMS} reliability from a failure mechanisms perspective, Microelectronics Reliability 43 (7) (2003) 1049 – 1060.

[24] J. Iannacci, Reliability of mems: A perspective on failure mechanisms, improvement solutions and best practices at development level, Displays 37 (2015) 62 – 71, advanced {MEMS} technologies and Displays.

[25] T. Weller, B. Hajek, Scheduling nonuniform traffic in a packet-switching system with small propagation delay, IEEE/ACM Transactions on Networking 5 (6) (1997) 813–823.

[26] C.-S. Chang, W.-J. Chen, H.-Y. Huang, Birkhoff-von Neumann input buffered crossbar switches, in: IEEE INFOCOM, 2000, pp. 1614–1623.

[27] M. Neely, E. Modiano, Y.-S. Cheng, Logarithmic delay for $n \times n$ packet switches under the crossbar constraint, IEEE/ACM Transactions on Networking 15 (3) (2007) 657 –668.

[28] H. Attiya, D. Hay, I. Keslassy, Packet-mode emulation of output-queued switches, in: SPAA, 2006, pp. 138–147.

[29] V. V. Vazirani, Approximation Algorithms, Springer, 2004.

[30] D. Aldous, Probability Approximations Via The Poisson Clumping Heuristic, Springer, 1989.

[31] S. Kandula, J. Padhye, P. Bahl, Flyways to de-congest data center networks (2009).

[32] B. Towles, W. Dally, Guaranteed scheduling for switches with configuration overhead, IEEE/ACM Transactions on Networking 11 (5) (2003) 835–847.

[33] S. Kotz, S. Nadarajah, Extreme value distributions: theory and applications, Imperial College Press, 2000.

Appendix A. Results from extreme value theory

We obtain here some results based on classical extreme value theory [30]. The aim of this theory is to evaluate the properties of the maximum among N i.i.d. random variables. Let γ be the Euler constant ($\gamma \approx 0.58$).

Lemma 1 (Bimodal case)**.** *Consider a set of N i.i.d. random variables $\{X_i\}_{i=1}^N$, in which $X_i = 0$ with probability p and $X_i = U_i$ with probability $1-p$, where U_i is a random variable exponentially distributed with average $1/\lambda$. Then, for $N \to \infty$:*

$$E\left[\max_{i=1,\ldots,N} X_i\right] \to \frac{1}{\lambda}(\gamma + \log N + \log(1-p))$$

Proof. Let Y be the random variable corresponding to the maximum among N samples: $Y = \max_{i=1,\ldots,N}\{X_i\}$. Following standard methodology, the corresponding cumulative distribution function (CDF) of Y can be obtained as follows:

$$\begin{aligned} F_Y(y) &= P(Y \le y) = P\left(\max_{i=1,\ldots,N}\{X_i\} \le y\right) \\ &= \prod_{i=1}^{N} P(X_i \le y) = \prod_{i=1}^{N} F_X(y) = F_X(y)^N \end{aligned}$$

Given the definition of X and recalling the CDF of an exponential distribution:

$$F_X(x) = p + (1-p)(1-e^{-\lambda x}) = 1 - (1-p)e^{-\lambda x}$$

for $x \ge 0$ and then

$$F_Y(x) = \left(1 - (1-p)e^{-\lambda x}\right)^N \tag{A.1}$$

Apply the following change of variable:

$$(1-p)e^{-\lambda x} = \frac{e^{-y}}{N}$$

then

$$x = \frac{1}{\lambda}(y + \log N + \log(1-p))$$

Exploiting (A.1):

$$F_Y\left(\frac{y + \log N + \log(1-p)}{\lambda}\right) = \left(1 - \frac{e^{-y}}{N}\right)^N$$

For $N \to \infty$:

$$\left(1 - \frac{e^{-y}}{N}\right)^N \to e^{-e^{-y}}$$

and

$$P(\lambda Y - \log N - \log(1-p) \leq y) = e^{-e^{-y}}$$

for $-\infty < y < +\infty$. After defining

$$Z = \lambda Y - \log N - \log(1-p) \tag{A.2}$$

we obtain $P(Z \leq y) = e^{-e^{-y}}$, which corresponds to the Gumbel-type distribution [33] whose average is the Euler constant γ. Hence, by combining (A.2) with $E[Z] = \gamma$, for $N \to \infty$ and, finally, we get our claim: $E[Y] \to (\gamma + \log N + \log(1-p))/\lambda$. □

Lemma 2 (Exponential case)**.** *Consider a set of N i.i.d. random variables $\{X_i\}_{i=1}^N$. If all X_i are exponentially distributed with average $1/\lambda$, then*

$$E\left[\max_{i=1,\ldots,N} X_i\right] \to \frac{1}{\lambda}(\gamma + \log N) \qquad \text{for } N \to \infty \tag{A.3}$$

Proof. The bimodal case for $p = 0$ corresponds to the exponential case. Just apply Lemma 1 to get the assert. □

Lemma 3 (Gaussian case)**.** *Consider a set of N i.i.d. random variables $\{X_i\}_{i=1}^N$. If all X_i have normal distribution with average a and variance b^2: $X \sim \mathcal{N}(a, b^2)$, then, for $N \to \infty$: $E\left[\max_{i=1,\ldots,N} X_i\right] \to a + b\Gamma(N)$, where $\Gamma(N)$ is defined as*

$$\Gamma(N) = (2\log N)^{\frac{1}{2}} - \frac{1}{2}(2\log N)^{-\frac{1}{2}}(\log(4\pi) + \log\log N) \tag{A.4}$$

Proof. The proof can be found in [30]. □